

Team Semantics and Recursive Enumerability

Antti Kuusisto

University of Wrocław and Technical University of Denmark

Abstract

It is well known that dependence logic captures the complexity class NP, and it has recently been shown that inclusion logic captures P on ordered models. These results demonstrate that team semantics offers interesting new possibilities for descriptive complexity theory. In order to properly understand the connection between team semantics and descriptive complexity, we introduce an extension D^* of dependence logic that can define exactly all recursively enumerable classes of finite models. Thus D^* provides an approach to computation alternative to Turing machines. The essential novel feature in D^* is an operator that can extend the domain of the considered model by a finite number of fresh elements. Due to the close relationship between generalized quantifiers and oracles, we also investigate generalized quantifiers in team semantics. We show that monotone quantifiers of type (1) can be canonically eliminated from quantifier extensions of first-order logic by introducing corresponding generalized dependence atoms.

1 Introduction

In this article we study logics based on *team semantics*. Team semantics was originally conceived by Hodges [10] in the context of IF-logic [9]. On the intuitive level, team semantics provides an alternative compositional approach to systems based on game-theoretic semantics. The compositional approach simplifies the more traditional game-theoretic approaches in several ways.

In [14], Väänänen introduced *dependence logic* (D), which is a novel approach to IF-logic based on new atomic formulae $=(x_1, \dots, x_k, y)$ that can be interpreted to mean that the choice for the value of y is *functionally determined* by the choices for the values of x_1, \dots, x_k in a semantic game.

After the introduction of dependence logic, research on logics based on team semantics has been *very active*. Several different logics with different applications have been investigated. Currently the two most important systems studied in the field in addition to dependence logic are *independence logic* [7] of Grädel and Väänänen and *inclusion logic* [5] of Galliani. Independence logic is a variant of dependence logic that extends first-order logic by new atomic formulae $x_1, \dots, x_k \perp y_1, \dots, y_k$ with the intuitive meaning that the interpretations of the variables x_1, \dots, x_k are *independent* of the interpretations of the variables y_1, \dots, y_k . Inclusion logic extends first-order logic by atomic formulae $x_1, \dots, x_k \subseteq y_1, \dots, y_k$, whose intuitive meaning is that each tuple interpreting the variables x_1, \dots, x_k must also be a tuple that interprets y_1, \dots, y_k . Exclusion logic, also introduced in [5] by Galliani, is a natural counterpart of inclusion

logic with atoms $x_1, \dots, x_k \mid y_1, \dots, y_k$ which state that the set of tuples interpreting x_1, \dots, x_k must not overlap with the set of tuples interpreting y_1, \dots, y_k .

It was observed in [14] and [7] that dependence logic and independence logic are both equi-expressive with *existential second-order logic*, and thereby capture NP. Curiously, it was established in [6] that inclusion logic is equi-expressive with *greatest fixed point logic* and thereby captures P on finite ordered models. These results show that team semantics offers a novel interesting perspective on descriptive complexity theory. Especially the very close connection between team semantics and game-theoretic concepts is interesting in this context.

In order *properly understand* the perspective on descriptive complexity provided by team semantics, it makes sense to accomodate the related logics in a unified umbrella framework that exactly characterizes the computational capacity of Turing machines. It turns out that there exists a particularly simple extension of dependence logic that does the job. Let D^* denote the logic obtained by extending first-order logic by the atoms of dependence, independence, inclusion, and exclusion logic, and furthermore, an operator I_x that extends the domain of the model considered by a finite number of *fresh* elements. We show below that D^* can define exactly all recursively enumerable classes of finite models.

Since D^* captures RE, it is not only a logic but also a model of computation. The striking simplicity of D^* and the link between team semantics and game-theory make D^* a particularly interesting system. There of course exist other logical frameworks where RE can be easily captured, such as abstract state machines [8], [1] and the recursive games of [12]. However, D^* provides a *simple unified perspective on recent advances in descriptive complexity based on team semantics*. The framework of [12] resembles D^* since it provides a perspective on RE that explains computational notions via game-theoretic concepts, but the approach in [12] uses potentially infinite games (and the article [12] also lacks a compositional approach). The approach provided by D^* is different.

The notion of a *generalized quantifier* can be seen as a natural logical counterpart of the computationally motivated notion of an *oracle*. Generalized quantifiers have recently been studied from the point of view of team semantics in [2], [3], [4], [11]. The focus has been on *monotone* quantifiers. We establish that type (1) monotone generalized quantifiers can be canonically eliminated from extensions of first-order logic by introducing corresponding *generalized dependence atoms*. This result demonstrates that team semantics indeed provides a natural approach to descriptive complexity theory and computation in general.

2 Preliminaries

We consider only models with a *purely relational vocabulary*, i.e., a vocabulary consisting of relation symbols only. Therefore, all vocabularies are below assumed to be purely relational without further warning. We let \mathfrak{A} , \mathfrak{B} , \mathfrak{C} , etc., denote models; A , B and C denote the domains of the models \mathfrak{A} , \mathfrak{B} and \mathfrak{C} , respectively.

We let VAR denote a countably infinite set of exactly all first-order *variable symbols*. Let $X \subseteq \text{VAR}$ be a *finite*, possibly empty set. Let A be a set. A function $s : X \rightarrow A$ is called an *assignment* with domain X and codomain A . We let $s[a/x]$ denote the assignment with domain $X \cup \{x\}$ and codomain $A \cup \{a\}$

defined such that $s[a/x](y) = a$ if $y = x$, and $s[a/x](y) = s(a)$ if $y \neq x$. Let T be a set. We define $s[T/x] = \{ s[a/x] \mid a \in T \}$.

Let $X \subseteq \text{VAR}$ be a finite, possibly empty set. Let U be a set of assignments $s : X \rightarrow A$. Such a set U is a *team* with *domain* X and *codomain* A . Note that the empty set is a team with codomain A , as is the set $\{\emptyset\}$ containing only the empty assignment. The team \emptyset does not have a unique domain; any finite subset of VAR is a domain of \emptyset . The domain of the team $\{\emptyset\}$ is \emptyset . The domain of team U is denoted by $\text{Dom}(U)$. Let T be a set. We define $U[T/x] := \{ s[a/x] \mid a \in T, s \in U \}$. Let $f : U \rightarrow \mathcal{P}(T)$ be a function, where \mathcal{P} denotes the power set operator. We define $U[f/x] := \bigcup_{s \in U} s[f(s)/x]$.

Let V be a team. Let $k \in \mathbb{Z}_+$, where \mathbb{Z}_+ denotes the positive integers. Let $x_1, \dots, x_k \in \text{Dom}(V)$. Define $\text{Rel}(V, (x_1, \dots, x_k)) := \{(s(x_1), \dots, s(x_k)) \mid s \in V\}$.

We then define *lax team semantics* for formulae of first-order logic (FO). As usual in investigations related to team semantics, formulae are assumed to be in *negation normal form*, i.e., negations occur only in front of atomic formulae. Let \mathfrak{A} be a model and U a team with codomain A . Let \models_{FO} denote the ordinary Tarskian satisfaction relation of first-order logic, i.e., $\mathfrak{A}, s \models_{\text{FO}} \varphi$ means that the model \mathfrak{A} satisfies the first-order formula φ under the assignment s . We define

$$\begin{array}{ll}
\mathfrak{A}, U \models x = y & \Leftrightarrow \forall s \in U (\mathfrak{A}, s \models_{\text{FO}} x = y), \\
\mathfrak{A}, U \models \neg x = y & \Leftrightarrow \forall s \in U (\mathfrak{A}, s \models_{\text{FO}} \neg x = y), \\
\mathfrak{A}, U \models R(x_1, \dots, x_k) & \Leftrightarrow \forall s \in U (\mathfrak{A}, s \models_{\text{FO}} R(x_1, \dots, x_k)), \\
\mathfrak{A}, U \models \neg R(x_1, \dots, x_k) & \Leftrightarrow \forall s \in U (\mathfrak{A}, s \models_{\text{FO}} \neg R(x_1, \dots, x_k)), \\
\mathfrak{A}, U \models (\varphi \wedge \psi) & \Leftrightarrow \mathfrak{A}, U \models \varphi \text{ and } \mathfrak{A}, U \models \psi, \\
\mathfrak{A}, U \models (\varphi \vee \psi) & \Leftrightarrow \mathfrak{A}, U_0 \models \varphi \text{ and } \mathfrak{A}, U_1 \models \psi \text{ for some} \\
& \text{teams } U_0, U_1 \subseteq U \text{ such that } U_0 \cup U_1 = U, \\
\mathfrak{A}, U \models \forall x \varphi & \Leftrightarrow \mathfrak{A}, U[A/x] \models \varphi, \\
\mathfrak{A}, U \models \exists x \varphi & \Leftrightarrow \mathfrak{A}, [f/x] \models \varphi \text{ for some } f : U \rightarrow (\mathcal{P}(A) \setminus \emptyset).
\end{array}$$

A sentence φ is true in \mathfrak{A} ($\mathfrak{A} \models \varphi$) if $\mathfrak{A}, \{\emptyset\} \models \varphi$. It is well known and easy to show that for an FO-formula φ , we have $\mathfrak{A}, U \models \varphi$ iff $\mathfrak{A}, s \models_{\text{FO}} \varphi$ for all $s \in U$.

Let (i_1, \dots, i_n) be a non-empty sequence of positive integers. A *generalized quantifier* of the type (i_1, \dots, i_n) is a class \mathcal{C} of structures (A, B_1, \dots, B_n) such that the following conditions hold.

1. $A \neq \emptyset$, and for each $j \in \{1, \dots, n\}$, we have $B_j \subseteq A^{i_j}$.
2. If $(A', B'_1, \dots, B'_n) \in \mathcal{C}$, and if there is an isomorphism $f : A' \rightarrow A''$ from (A', B'_1, \dots, B'_n) to $(A'', B''_1, \dots, B''_n)$, then $(A'', B''_1, \dots, B''_n) \in \mathcal{C}$.

Let Q be a quantifier of the type (i_1, \dots, i_n) . Let $A \neq \emptyset$ be a set. We define Q^A to be the set $\{(B_1, \dots, B_n) \mid (A, B_1, \dots, B_n) \in Q\}$. If Q is a quantifier of type (1), we define $Q' := \{(A, B) \mid (A, A \setminus B) \in Q\}$.

A quantifier Q of type (1) is said to be *monotone*, if the condition $A \subseteq B \Rightarrow (A \in Q^C \Rightarrow B \in Q^C)$ holds for all $C \neq \emptyset$ and $A, B \subseteq C$.

Let us next see how the ordinary Tarskian semantic relation \models_{FO} is extended to deal with languages with generalized quantifiers of type (1). Let Q be a quantifier of type (1). Let $\text{FO}(Q)$ denote the extension of FO obtained by adding a new formula formation rule that constructs $Qx\varphi$ from φ . Let \mathfrak{A} be a model and s an assignment with codomain A . We define $\mathfrak{A}, s \models_{\text{FO}} Qx\varphi$ iff we have $\{a \in A \mid \mathfrak{A}, s[a/x] \models_{\text{FO}} \varphi\} \in Q^A$. Note that the formula $\neg Qx\varphi$ is

equivalent to $Q^d x \neg \varphi$, where $(\cdot)^d$ denotes the *dualizing operation* defined such that $Q^d := \{ (C, D) \mid (C, C \setminus D) \notin Q \}$. Note also that $(Q^d)^d = Q$, and that if Q is monotone, then so is Q^d . Thus a language of first-order logic extended with monotone type (1) quantifiers can be represented in negation normal form such that the resulting language is also essentially an extension of FO by monotone type (1) quantifiers.

We next show how to extend lax team semantics to first-order logic with monotone generalized quantifiers of type (1). (*We investigate only quantifiers of type (1) for the sake of simplicity and brevity; a somewhat more general approach will be taken up in the journal version.*) As usual, formulae are taken to be in negation normal form. We define (cf. [2]) that $\mathfrak{A}, U \models Qx \varphi$ iff there exists a function $f : U \rightarrow Q^A$ such that $\mathfrak{A}, U[f/x] \models \varphi$. The following proposition from [2] is straightforward to prove by induction on the structure of formulae.

Proposition 2.1. *Let φ be a formula of first-order logic extended with generalized quantifiers. Let U be a team. Then $\mathfrak{A}, U \models \varphi$ iff $\forall s \in U (\mathfrak{A}, s \models_{\text{FO}} \varphi)$.*

Let $n \in \mathbb{Z}_+$. Let Q be a generalized quantifier of type (i_1, \dots, i_n) . Extend the syntax of first-order logic with atomic expressions $A_Q(\bar{x}_1, \dots, \bar{x}_n)$, where each \bar{x}_j is a tuple of variables of length i_j . (*Negated generalized atoms are not allowed, and we only consider logics in negation normal form.*) Let U be a team whose domain contains all variables that occur in the tuples $\bar{x}_1, \dots, \bar{x}_n$. We extend the lax team semantics defined above such that $\mathfrak{A}, U \models A_Q(\bar{x}_1, \dots, \bar{x}_n)$ iff $(\text{Rel}(U, \bar{x}_1), \dots, \text{Rel}(U, \bar{x}_n)) \in Q^A$. The generalized quantifier Q defines a *generalized atom* A_Q of the type (i_1, \dots, i_n) . This definition is from [11].

Let $k \in \mathbb{Z}_+$. Let D_k denote the generalized quantifier that contains exactly all structures (A, R) such that $A \neq \emptyset$ and $R \subseteq A^k$ satisfies the condition that if $(a_1, \dots, a_{k-1}, b) \in R$ and $(a_1, \dots, a_{k-1}, c) \in R$, then we have $b = c$. *Dependence logic* (D) is the extension of first-order logic in negation normal form with the generalized atoms A_{D_k} for each positive integer k . These atoms are called *dependence atoms*. Below we will write $\equiv(x_1, \dots, x_k)$ instead of $A_{D_k}(x_1, \dots, x_k)$. We note that dependence logic is sometimes formulated such that negated atoms $\neg \equiv(x_1, \dots, x_k)$ are allowed, but since the semantics then dictates that $\mathfrak{A}, U \models \neg \equiv(x_1, \dots, x_k)$ iff $U = \emptyset$, these negated atoms can be replaced by $\exists x(x \neq x)$.

Inclusion logic is obtained by extending first-order logic in negation normal form by atoms $x_1, \dots, x_k \subseteq y_1, \dots, y_k$ with the semantics $\mathfrak{A}, U \models x_1, \dots, x_k \subseteq y_1, \dots, y_k$ iff $\text{Rel}(U, (x_1, \dots, x_k)) \subseteq \text{Rel}(U, (y_1, \dots, y_k))$. Here k can be any positive integer. Similarly, *exclusion logic* extends first-order logic in negation normal form with atoms $x_1, \dots, x_k \mid y_1, \dots, y_k$ such that $\mathfrak{A}, U \models x_1, \dots, x_k \mid y_1, \dots, y_k$ iff $\text{Rel}(U, (x_1, \dots, x_k)) \cap \text{Rel}(U, (y_1, \dots, y_k)) = \emptyset$. Again k can be any positive integer. *Independence logic* extends first-order logic in negation normal form with atoms $x_1, \dots, x_k \perp_{z_1, \dots, z_m} y_1, \dots, y_n$ such that $\mathfrak{A}, U \models x_1, \dots, x_k \perp_{z_1, \dots, z_m} y_1, \dots, y_n$ iff for all $s, s' \in U$ there exists a $t \in U$ such that

$$\left(\bigwedge_{i \leq m} s(z_i) = s'(z_i) \right) \Rightarrow \left(\bigwedge_{i \leq k} t(x_i) = s(x_i) \wedge \bigwedge_{i \leq m} t(z_i) = s(z_i) \wedge \bigwedge_{i \leq n} t(y_i) = s'(y_i) \right).$$

Here k, m, n can be any positive integers. Independence logic also contains atoms $x_1, \dots, x_k \perp y_1, \dots, y_n$ such that $\mathfrak{A}, U \models x_1, \dots, x_k \perp y_1, \dots, y_n$ iff for all $s, s' \in U$ there exists a $t \in U$ such that $\bigwedge_{i \leq k} t(x_i) = s(x_i)$ and $\bigwedge_{i \leq n} t(y_i) = s'(y_i)$. Here k and n can be any positive integers.

Let \mathfrak{A} be a model and τ its vocabulary. Let $S \neq \emptyset$ be a set such that $S \cap A = \emptyset$. We let $\mathfrak{A} + S$ denote the model \mathfrak{B} such that $B = A \cup S$ and $R^{\mathfrak{B}} = R^{\mathfrak{A}}$ for all $R \in \tau$. The model \mathfrak{B} is called a *finite bloating* of \mathfrak{A} .

We then define the logic D^* that captures recursive enumerability. In the spirit of team semantics, D^* is based on the use of sets of assignments, i.e., teams, that involve first-order variables. Let D^+ denote the logic obtained by extending first-order logic in negation normal form by all dependence atoms, independence atoms, inclusion atoms, and exclusion atoms. D^* is obtained by extending D^+ by an additional formula formation rule stating that if φ is a formula, then so is $\text{Ix } \varphi$. We define $\mathfrak{A}, U \models \text{Ix } \varphi$ iff there exists a finite bloating $\mathfrak{A} + S$ of \mathfrak{A} such that $\mathfrak{A} + S, U[S/x] \models \varphi$. We note that since $\perp(x_1, \dots, x_k, y)$ is equivalent to $y \perp_{x_1, \dots, x_k} y$, dependence atoms can in fact be eliminated from D^* .

Note that if desired, we can avoid reference to a proper class of possible bloatings of \mathfrak{A} by letting $A_1 := A \cup \{A\}$ to be the canonical bloating of A by one element and $A_{k+1} := A_k \cup \{A_k\}$ the bloating of A by $k+1$ elements.

3 D^* Captures RE

Let τ be a vocabulary. Sentences of *existential second-order logic* (ESO) over τ are formulae of the type $\exists X_1 \dots \exists X_k \varphi$, where X_1, \dots, X_k are relation variables and φ a sentence of FO over $\tau \cup \{X_1, \dots, X_k\}$. The symbols X_1, \dots, X_k are not in τ . We extend ESO by defining a logic \mathcal{L}_{RE} , whose τ -sentences are of the type $\text{IY } \psi$, where $Y \notin \tau$ is a *unary* relation variable and ψ an ESO-sentence over $\tau \cup \{Y\}$. Let \mathfrak{A} be a τ -model. The semantics of \mathcal{L}_{RE} is defined such that $\mathfrak{A} \models \text{IY } \psi$ iff there exists a finite set $S \neq \emptyset$ such that the following conditions hold.

1. $A \cap S = \emptyset$.
2. Let \mathfrak{A}^+ be the model of the vocabulary $\tau \cup \{Y\}$ with domain $A \cup S$ such that $Y^{\mathfrak{A}^+} = S$ and $R^{\mathfrak{A}^+} = R^{\mathfrak{A}}$ for all $R \in \tau$. We have $\mathfrak{A}^+ \models \psi$.

As we shall see, the logic \mathcal{L}_{RE} can define in the finite exactly all recursively enumerable classes of finite models.

Let $\sigma \neq \emptyset$ be a finite set of unary relation symbols and *Succ* a binary relation symbol. A *word model* over the vocabulary $\{\text{Succ}\} \cup \sigma$ is a model \mathfrak{A} defined as follows.

1. The domain A of \mathfrak{A} is a nonempty finite set. The predicate *Succ* is a successor relation over A , i.e., a binary relation corresponding to a linear order, but with maximum out-degree and in-degree equal to one.
2. Let $b \in A$ be the smallest element with respect to *Succ*. We have $b \notin P^{\mathfrak{A}}$ for all $P \in \sigma$. (This is because we do not allow models with the empty domain; the empty word corresponds to the word model with exactly one element.) For all $a \in A \setminus \{b\}$, there is exactly one $P \in \sigma$ such that $a \in P^{\mathfrak{A}}$.

Word models canonically encode finite words. For example the word *abbaa* over the alphabet $\{a, b\}$ is encoded by the word model \mathfrak{M} over the vocabulary $\{\text{Succ}, P_a, P_b\}$ defined such that $M = \{0, \dots, 5\}$ and *Succ* $^{\mathfrak{M}}$ is the canonical successor relation on M , and we have $P_a^{\mathfrak{M}} = \{1, 4, 5\}$ and $P_b^{\mathfrak{M}} = \{2, 3\}$.

When investigating computations on structure classes (rather than strings), Turing machines of course operate on *encodings* of structures. We will use the encoding scheme of [13]. Let τ be a finite vocabulary and \mathfrak{A} a finite τ -structure. In order to encode the structure \mathfrak{A} by a binary string, we first need to define a linear ordering of the domain A of \mathfrak{A} . Let $<^{\mathfrak{A}}$ denote such an ordering.

Let $R \in \tau$ be a k -ary relation symbol. The encoding $enc(R^{\mathfrak{A}})$ of $R^{\mathfrak{A}}$ is the $|A|^k$ -bit string defined as follows. Consider an enumeration of all k -tuples over A in the *lexicographic order* defined with respect to $<^{\mathfrak{A}}$. In the lexicographic order, (a_1, \dots, a_k) is smaller than (a'_1, \dots, a'_k) iff there exists $i \in \{1, \dots, k\}$ such that $a_i < a'_i$ and $a_j = a'_j$ for all $j < i$. There are $|A|^k$ tuples in A^k , and the string $enc(R^{\mathfrak{A}})$ is the word $t \in \{0, 1\}^*$ of the length $|A|^k$ such that the bit t_i of $t = t_1 \dots t_{|A|^k}$ is 1 if and only if the i -th tuple $(a_1, \dots, a_k) \in A^k$ in the lexicographic order is in the relation $R^{\mathfrak{A}}$.

The encoding $enc(\mathfrak{A})$ is defined as follows. We first order the relations in τ . Let p be the number of relations in τ , and let R_1, \dots, R_p enumerate the symbols in τ according to the order. We define $enc(\mathfrak{A}) := 0^{|A|} \cdot 1 \cdot enc(R_1^{\mathfrak{A}}) \cdot \dots \cdot enc(R_p^{\mathfrak{A}})$. Notice that the encoding of \mathfrak{A} indeed depends on the order $<^{\mathfrak{A}}$ and the ordering of the relation symbols in τ , so \mathfrak{A} in general has several encodings. However, we assume that τ is always ordered in some canonical way, so the multiplicity of encodings results in only due to different orderings of the domain of \mathfrak{A} .

Let τ be a finite vocabulary. A Turing machine TM defines a *semi-decision algorithm for a class \mathcal{C} of finite τ -models* iff there is an accepting run for TM on an input $w \in \{0, 1\}^*$ exactly when w is some encoding of some structure $\mathfrak{A} \in \mathcal{C}$.

Proposition 3.1. *In the finite, \mathcal{L}_{RE} can define exactly all recursively enumerable classes of models.*

Sketch. Let TM be a Turing machine that defines a semi-decision algorithm for some class of models. It is routine to write a formula $\varphi_{TM} := IY\overline{\exists X}\psi$ such that $\mathfrak{A} \models \varphi_{TM}$ iff there exists an extension \mathfrak{B} of \mathfrak{A} that consist essentially of a copy of \mathfrak{A} and another part \mathfrak{C} that encodes the computation table of an accepting computation of TM on an input $enc(\mathfrak{A})$. We can use the predicates in $\overline{\exists X}$ in order to define word models that encode $enc(\mathfrak{A})$ and other strings that correspond to the Turing machine tape at different stages of the computation. Symbols in $\overline{\exists X}$ can also be used, inter alia, in order to define the other parts of the computation table and an ordering of the domain of \mathfrak{A} , and also relations that connect \mathfrak{A} to \mathfrak{C} in order to ensure \mathfrak{A} and \mathfrak{C} are correctly related. The symbol Y is used in order to see which points belong to the original model \mathfrak{A} .

For the converse, given a sentence $IY\overline{\exists X}\psi$ of \mathcal{L}_{RE} , we can define a Turing machine that first non-deterministically provides a number $k \in \mathbb{Z}_+$ of fresh points to be added to the domain of the model considered, and then checks if $\overline{\exists X}\psi$ holds in the obtained larger model. \square

If desired, obviously \mathcal{L}_{RE} can be modified without change in expressivity such that the set of fresh points labelled by Y can also be possibly empty. We define \mathcal{L}_{RE} with Y always nonempty simply because of technical issues related to the treatment of disjunction in team semantics (see below). It is also worth noting here that \mathcal{L}_{RE} is a *rather straightforward characterization of RE* in terms of a logic that is almost classical. Indeed, \mathcal{L}_{RE} is rather similar to ESO.

Our next aim is to prove Lemma 3.2, which essentially provides a way of encoding a unary relation symbol Y by a corresponding variable symbol y with

the help of inclusion, exclusion, and independence atoms. For the purposes of the Lemma, we first define a translation from dependence logic to D^+ .

Let χ be a *sentence* of dependence logic over a vocabulary τ such that $Y \notin \tau$. Let y, v, u, u' be variables that *do not occur* in χ . We next define a translation $T_Y^y(\chi)$ of χ into D^+ by recursion on the structure of χ . (Strictly speaking, the variables v, u, u' are fixed parameters of the translation just like y and Y , so we should write $T_Y^{y,v,u,u'}(\chi)$ instead of $T_Y^y(\chi)$. The issue here is only that when a sentence χ is translated, the auxiliary variables y, v, u, u' should not occur in χ .)

1. $T_Y^y(R(x_1, \dots, x_k)) := R(x_1, \dots, x_k)$ and $T_Y^y(\neg R(x_1, \dots, x_k)) := \neg R(x_1, \dots, x_k)$
2. $T_Y^y(x = z) := x = z$ and $T_Y^y(\neg x = z) := \neg x = z$
3. $T_Y^y(Y(x)) := x \subseteq y$ and $T_Y^y(\neg Y(x)) := x|y$
4. $T_Y^y(=(x_1, \dots, x_k)) := =(x_1, \dots, x_k)$
5. $T_Y^y((\varphi \wedge \psi)) := ((T_Y^y(\varphi) \wedge T_Y^y(\psi)))$
6. $T_Y^y((\varphi \vee \psi)) := \exists v(v \perp_{\bar{z}} y \wedge ((T_Y^y(\varphi) \wedge v = u) \vee (T_Y^y(\psi) \wedge v = u')))$, where \bar{z} contains exactly all variables quantified superordinate to $(\varphi \vee \psi)$ in χ , i.e., exactly each x such that $(\varphi \vee \psi)$ is in the scope of $\exists x$ or $\forall x$.
7. Assume $\exists x \varphi$ is subordinate to a disjunction in χ , meaning that there is a subformula $(\alpha \vee \beta)$ of χ and $\exists x \varphi$ is a subformula of $(\alpha \vee \beta)$. We define $T_Y^y(\exists x \varphi) := \exists x(x \perp_{\bar{z}} y \wedge T_Y^y(\varphi))$, where \bar{z} contains exactly all variables quantified superordinate to $\exists x \varphi$ in χ , with the exception that \bar{z} never contains x ; the exception is relevant if χ contains nested quantification of x .
8. Assume $\exists x \varphi$ is not subordinate to a disjunction in χ . Then $T_Y^y(\exists x \varphi) := \exists x(x \perp_{\bar{z}} y \wedge T_Y^y(\varphi))$, where \bar{z} contains exactly all variables quantified superordinate to $\exists x \varphi$ in χ , with the exception that \bar{z} never contains x .
9. $T_Y^y(\forall x \varphi) := \forall x(T_Y^y(\varphi))$

Let \mathfrak{A} be a model such that $|A| \geq 2$. Let $S \subseteq A$. Let χ be a sentence of dependence logic and φ a subformula of χ . Let (U, V) a pair of be teams with codomain A such that the following conditions hold.

1. Call $Z := \text{Dom}(U)$. Z contains exactly all variables quantified superordinate to φ in χ . $\text{Dom}(V)$ is $Z \cup \{y, u, u'\}$ or $Z \cup \{y, v, u, u'\}$; we have $v \in \text{Dom}(V)$ iff φ is subordinate to a disjunction in χ .
2. We have $U = V \upharpoonright Z$, i.e., $U = \{s \upharpoonright Z \mid s \in V\}$, where $Z = \text{Dom}(U)$.
3. There exists a team X such that $V = X[S/y]$. (Thus $S \neq \emptyset$ if $V \neq \emptyset$.)
4. For all $s, t \in V$, we have $s(u) = t(u) \neq t(u') = s(u')$. In other words, every assignment in V gives exactly the same interpretation to u and to u' , and the interpretation of u is different from that of u' .

When (U, V) satisfies the above four conditions, we say that (U, V) is a *suitable pair* for \mathfrak{A} , $S \subseteq A$, (y, v, u, u') and (φ, χ) . Below \mathfrak{A} , S , and (y, v, u, u') will always be clear from the context (and in fact the same everywhere), so we may simply talk about suitable pairs for (φ, χ) .

Let \mathfrak{B} be a model and $T \subseteq B$ a set. Let τ be the vocabulary of \mathfrak{B} . Let $P \notin \tau$ be a unary relation symbol. We let $(\mathfrak{B}, P \mapsto T)$ denote the expansion of \mathfrak{B} to the vocabulary $\tau \cup \{P\}$ such that $P^{\mathfrak{B}} = T$.

Let s be an assignment with domain X . Let $\{x_1, \dots, x_k\}$ be a finite set of variables. We let $s_{-\{x_1, \dots, x_k\}}$ denote the assignment $s \upharpoonright (X \setminus \{x_1, \dots, x_k\})$.

Lemma 3.2. *Let χ be a sentence of dependence logic not containing the symbols y, v, u, u' . Let \mathfrak{A} be a model with at least two elements. Let Y be a unary relation symbol that occurs neither in χ nor in the vocabulary of \mathfrak{A} . Let $S \subseteq A$. Let $(\{\emptyset\}, V)$ be a suitable pair of for \mathfrak{A} , S , (y, v, u, u') and (χ, χ) . Then we have $(\mathfrak{A}, Y \mapsto S), \{\emptyset\} \models \chi$ iff $\mathfrak{A}, V \models T_Y^y(\chi)$.*

Proof. We prove by induction on the structure of χ that for any subformula φ of χ , the equivalence $(\mathfrak{A}, Y \mapsto S), U \models \varphi \Leftrightarrow \mathfrak{A}, V \models T_Y^y(\varphi)$ holds for all suitable pairs (U, V) for \mathfrak{A} , S , (y, v, u, u') and (φ, χ) . The cases for literals $R(x_1, \dots, x_k)$, $\neg R(x_1, \dots, x_k)$, $x = z$ and $\neg x = z$ are clear, as is the case for $=(x_1, \dots, x_k)$. The cases for literals $Y(x)$ and $\neg Y(x)$ follow directly by the definition of the atoms $x \subseteq y$ and $x|y$. The cases for $(\varphi \wedge \psi)$ and $\forall x \varphi$ follow easily by the semantics of \wedge and \forall and the induction hypothesis.

Let (U, V) be a suitable pair for \mathfrak{A} , S , (y, v, u, u') , and $(\varphi \vee \psi, \chi)$. Assume that $(\mathfrak{A}, Y \mapsto S), U \models \varphi \vee \psi$. Thus $(\mathfrak{A}, Y \mapsto S), U_0 \models \varphi$ and $(\mathfrak{A}, Y \mapsto S), U_1 \models \psi$ for some U_0 and U_1 such that $U_0 \cup U_1 = U$. We must show that

$$\mathfrak{A}, V \models \exists v (v \perp_{\bar{z}} y \wedge ((T_Y^y(\varphi) \wedge v = u) \vee (T_Y^y(\psi) \wedge v = u'))). \quad (1)$$

Define a function $g : V \rightarrow (\mathcal{P}(A) \setminus \{\emptyset\})$ so that the following conditions hold.

1. If $s_{-\{u, u', v, y\}} \in U_0 \setminus U_1$, then $g(s) = \{s(u)\}$.
2. If $s_{-\{u, u', v, y\}} \in U_1 \setminus U_0$, then $g(s) = \{s(u')\}$.
3. If $s_{-\{u, u', v, y\}} \in U_0 \cap U_1$, then $g(s) = \{s(u), s(u')\}$.

Call $Z := V[g/v]$. We must show that

$$\mathfrak{A}, Z \models v \perp_{\bar{z}} y \wedge ((T_Y^y(\varphi) \wedge v = u) \vee (T_Y^y(\psi) \wedge v = u')). \quad (2)$$

By the definition of g , it is clear that $\mathfrak{A}, Z \models v \perp_{\bar{z}} y$. To deal with the rest of Equation 2, we need to find teams Z_0 and Z_1 such that $\mathfrak{A}, Z_0 \models T_Y^y(\varphi) \wedge v = u$ and $\mathfrak{A}, Z_1 \models T_Y^y(\psi) \wedge v = u'$, and furthermore, $Z_0 \cup Z_1 = Z$.

Define $Z_0 := \{s \in Z \mid s(v) = s(u)\}$ and $Z_1 := \{s \in Z \mid s(v) = s(u')\}$. To see that (U_0, Z_0) and (U_1, Z_1) are suitable pairs for (φ, χ) and (ψ, χ) , respectively, we consider the pair (U_0, Z_0) . (The argument for (U_1, Z_1) is similar.)

It is clear that the domain of Z_0 is $\text{Dom}(U_0) \cup \{y, v, u, u'\}$. The fact that $U_0 = Z_0 \upharpoonright \text{Dom}(U_0)$ follows essentially by the definition of the function g . Also, g was defined such that for all $s \in V$, the set $g(s)$ is independent of $s(y)$, and furthermore, Z_0 was defined such that for all $s' \in Z$, whether or not $s' \in Z_0$ holds, is independent of $s'(y)$. Thus there exists a team X such that

$Z_0 = X[S/y]$. Finally, it is clear that $s(u) = t(u) \neq s(u') = t(u')$ holds for all $s, t \in Z_0$.

Thus (U_0, Z_0) is a suitable pair. A similar argument shows that also (U_1, Z_1) is a suitable pair. We know that $\mathfrak{A}, U_0 \models \varphi$, and since (U_0, Z_0) is a suitable pair, we may apply the induction hypothesis in order to conclude that $\mathfrak{A}, Z_0 \models T_Y^y(\varphi)$. Similarly, we conclude that $\mathfrak{A}, Z_1 \models T_Y^y(\psi)$. Due to the definition of Z_0 and Z_1 , it is clear that $Z = Z_0 \cup Z_1$, and also that $\mathfrak{A}, Z_0 \models v = u$ and $\mathfrak{A}, Z_1 \models v = u'$. Since we already saw that $\mathfrak{A}, Z \models v \perp_{\bar{z}} y$, we conclude that Equation 2 holds.

For the converse, we assume that Equation 1 holds and show that $(\mathfrak{A}, Y \mapsto S), U \models (\varphi \vee \psi)$. Here (U, V) is a suitable pair for $\mathfrak{A}, S, (y, v, u, u')$, and $(\varphi \vee \psi, \chi)$. By Equation 1, there exists a function $h : V \rightarrow (\mathcal{P}(A) \setminus \emptyset)$ such that we have $\mathfrak{A}, V[h/v] \models v \perp_{\bar{z}} y \wedge ((T_Y^y(\varphi) \wedge v = u) \vee (T_Y^y(\psi) \wedge v = u'))$. Call $W := V[h/v]$. Thus there exist teams W_0 and W_1 such that

$$\mathfrak{A}, W_0 \models (T_Y^y(\varphi) \wedge v = u) \text{ and } \mathfrak{A}, W_1 \models (T_Y^y(\psi) \wedge v = u'), \quad (3)$$

and furthermore $W = W_0 \cup W_1$. Define $U_0 = W_0 \upharpoonright \text{Dom}(U)$ and $U_1 = W_1 \upharpoonright \text{Dom}(U)$. Since $\mathfrak{A}, W \models v \perp_{\bar{z}} y$, there exists a team X such that $W = X[S/y]$. Thus, since $\mathfrak{A}, W_0 \models v = u$ and $\mathfrak{A}, W_1 \models v = u'$, there exist teams X_0 and X_1 such that $W_0 = X_0[S/y]$ and $W_1 = X_1[S/y]$. Hence (U_0, W_0) and (U_1, W_1) are suitable pairs for (φ, χ) and (ψ, χ) , respectively. Thus we may apply the induction hypothesis in order to conclude by Equation 3 that $(\mathfrak{A}, Y \mapsto S), U_0 \models \varphi$ and $(\mathfrak{A}, Y \mapsto S), U_1 \models \psi$. As clearly $U_0 \cup U_1 = U$, we have $(\mathfrak{A}, Y \mapsto S), U \models (\varphi \vee \psi)$.

We then discuss formulae of the type $\exists x \varphi$. We consider the case where $\exists x \varphi$ is subordinate to a disjunction; the case where this does not hold is similar. Let (U, V) be a suitable pair for $(\exists x \varphi, \chi)$. Assume that $(\mathfrak{A}, Y \mapsto S), U \models \exists x \varphi$. Thus there exists a function $f : U \rightarrow (\mathcal{P}(A) \setminus \emptyset)$ such that $(\mathfrak{A}, Y \mapsto S), U[f/x] \models \varphi$. Define a function $g : V \rightarrow (\mathcal{P}(A) \setminus \emptyset)$ such that $g(s) = f(s_{-\{y, u, u', v\}})$ for all $s \in V$. Clearly we have $\mathfrak{A}, V[g/x] \models x \perp_{\bar{z}} yv$, and clearly $(U[f/x], V[g/x])$ is a suitable pair. Therefore, as $(\mathfrak{A}, Y \mapsto S), U[f/x] \models \varphi$, we conclude that $\mathfrak{A}, V[g/x] \models T_Y^y(\varphi)$ by the induction hypothesis. Thus $\mathfrak{A}, V[g/x] \models x \perp_{\bar{z}} yv \wedge T_Y^y(\varphi)$, and therefore $\mathfrak{A}, V \models \exists x(x \perp_{\bar{z}} yv \wedge T_Y^y(\varphi))$.

For the converse, assume that $\mathfrak{A}, V \models \exists x(x \perp_{\bar{z}} yv \wedge T_Y^y(\varphi))$. Thus there exists a function $h : V \rightarrow (\mathcal{P}(A) \setminus \emptyset)$ such that $\mathfrak{A}, V[h/x] \models x \perp_{\bar{z}} yv \wedge T_Y^y(\varphi)$. Let $F : U \rightarrow V$ be a function such that for each $s \in U$, we have $s = (F(s))_{-\{y, u, u', v\}}$. Define the function $k : U \rightarrow (\mathcal{P}(A) \setminus \emptyset)$ such that $k(s) = h(F(s))$ for all $s \in U$. As $\mathfrak{A}, V[h/x] \models x \perp_{\bar{z}} yv$, we see that $(U[k/x], V[h/x])$ is a suitable pair. Thus we may use the induction hypothesis to conclude that, since $\mathfrak{A}, V[h/x] \models T_Y^y(\varphi)$, we have $(\mathfrak{A}, Y \mapsto S), U[k/x] \models \varphi$. Thus $(\mathfrak{A}, Y \mapsto S), U \models \exists x \varphi$. \square

Define $\mathbb{T}_Y^y(\varphi) := \exists u \exists u' (u \neq u' \wedge = (u) \wedge = (u') \wedge T_Y^y(\varphi))$. The following Lemma now follows directly.

Lemma 3.3. *Let \mathfrak{A} be a model such that $|A| \geq 2$. Let $S \subseteq A$ be a nonempty finite set. Let φ be a sentence of dependence logic. Let y be a variable that does not occur in φ . Let Y be a unary symbol that occurs neither in φ nor in the vocabulary of \mathfrak{A} . Then $(\mathfrak{A}, Y \mapsto S), \{\emptyset\} \models \varphi$ iff $\mathfrak{A}, \{\emptyset\}[S/y] \models \mathbb{T}_Y^y(\varphi)$.*

Theorem 3.4. \mathcal{L}_{RE} is contained in D^* .

Proof. It is well known that every sentence α of ESO translates to an equivalent sentence $\alpha^\#$ of dependence logic, see [14]. We shall use this translation below.

Let $\varphi := \text{IY } \overline{\exists X} \psi$ be a sentence of \mathcal{L}_{RE} , where ψ is a first-order sentence. The following chain of equivalences, where the penultimate equivalence follows by Lemma 3.3, settles the current theorem.

$$\begin{aligned} \mathfrak{A} \models \varphi &\Leftrightarrow (\mathfrak{A} + S, Y \mapsto S) \models \overline{\exists X} \psi \text{ for some finite } S \neq \emptyset \text{ s.t. } S \cap A = \emptyset \\ &\Leftrightarrow (\mathfrak{A} + S, Y \mapsto S), \{\emptyset\} \models (\overline{\exists X} \psi)^\# \text{ for some finite } S \neq \emptyset \text{ s.t. } S \cap A = \emptyset \\ &\Leftrightarrow \mathfrak{A} + S, \{\emptyset\}[S/y] \models \mathbb{T}_Y^y((\overline{\exists X} \psi)^\#) \text{ for some finite } S \neq \emptyset \text{ s.t. } S \cap A = \emptyset \\ &\Leftrightarrow \mathfrak{A}, \{\emptyset\} \models \text{Iy } \mathbb{T}_Y^y((\overline{\exists X} \psi)^\#) \end{aligned}$$

□

Theorem 3.5. D^* is contained in \mathcal{L}_{RE} .

Proof. Let φ be a sentence of D^* . Assume φ contains k occurrences of the operator I . Let TM be a Turing machine such that when given an input model \mathfrak{A} , TM first nondeterministically constructs a tuple $\overline{n} \in (\mathbb{Z}_+)^k$ that gives for each occurrence of I in φ a number of new points to be added to the model. Then TM checks whether \mathfrak{A} satisfies φ with the given tuple \overline{n} of cardinalities to be added during the evaluation. TM is a semi-decision algorithm corresponding to φ . □

4 Eliminating Monotone Generalized Quantifiers

In this section we show that monotone quantifiers of type (1) can be eliminated from quantifier extensions of FO by introducing generalized atoms that are canonically similar to the quantifiers.

Let Q be a monotone type (1) quantifier and \overline{y} a tuple of variables of length $k \in \mathbb{N}$. We let $A_Q(\overline{y}, x)$ denote the type $(k+1)$ atom defined such that $\mathfrak{A}, U \models A_Q(\overline{y}, x)$ iff $\text{Rel}(U_0, x) \in Q^A$ for each maximal nonempty team $U_0 \subseteq U$ with the property $\forall s, t \in U_0 (s(\overline{y}) = t(\overline{y}))$. We call this atom the *k-atom induced by Q*.

Let φ be a formula of FO, possibly extended with monotone quantifiers of type (1). Recall from Section 2 that $\neg Qx \varphi$ is equivalent to $Q^d x \neg \varphi$, and if Q is a monotone type (1) quantifier, then so is Q^d .

Let Q be a monotone type (1) quantifier. We let $\text{FO}(Q)$ denote first-order logic in negation normal form and extended by the quantifiers Q and Q^d . We let $\text{FO}(\mathcal{A}_Q)$ denote first-order logic in negation normal form and extended by all k -atoms induced by Q and Q' for all $k \in \mathbb{N}$. We define a translation of formulae of $\text{FO}(Q)$ into $\text{FO}(\mathcal{A}_Q)$ as follows.

Let α be a formula of $\text{FO}(Q)$. First, if necessary, rename bound variables of α so that no free variable is also a bound variable. Then remove all nested quantification of the same variable by renaming variables; this means that in the resulting formula, no quantifier quantifying x is not allowed to be in the scope of another quantifier quantifying the same variable x . We call the resulting formula *clean*. We then translate the *clean* formula β to a formula β^* , where $(\cdot)^*$ is defined by the rules below. Note that \exists is a type (1) monotone generalized quantifier and $\exists^d = \forall$, so the case for Q below covers also \exists and \forall .

1. For atoms and negated atoms, $\varphi^* := \varphi$.

2. $((\varphi \wedge \psi))^* := (\varphi^* \wedge \psi^*)$ and $((\varphi \vee \psi))^* := (\varphi^* \vee \psi^*)$.
3. $(Qx\varphi)^* := \forall x((A_Q(\overline{y}, x) \wedge \varphi^*) \vee A_{Q'}(\overline{y}, x))$, where \overline{y} contains exactly all variables quantified superordinate to $Qx\varphi$ in the original clean formula β , and also the free variables of β .

Note that, interestingly, we eliminated the use of \exists altogether.

Theorem 4.1. *Let Q be a generalized quantifier of type (1). Every formula of χ of $\text{FO}(Q)$ can be effectively translated into a formula χ^* of $\text{FO}(\mathcal{A}_Q)$ such that $\mathfrak{A}, U \models \chi \Leftrightarrow \mathfrak{A}, U \models \chi^*$ for all \mathfrak{A} and U .*

Proof. Let β be a clean formula of $\text{FO}(Q)$. We will show by induction on the structure of β that $\mathfrak{A}, U \models \varphi \Leftrightarrow \mathfrak{A}, U \models \varphi^*$ for all subformulae φ of β .

The cases for atomic and negated atomic formulae are clear, as are the cases for \wedge and \vee . We discuss the quantifier case in detail. Assume $\mathfrak{A}, U \models Qx\varphi$. We assume that $U \neq \emptyset$; the case for $U = \emptyset$ is straightforward, as long as it is kept in mind that \emptyset is a function with domain \emptyset . Let $s \in U$ be an arbitrary assignment. By Proposition 2.1, we have $\mathfrak{A}, s \models_{\text{FO}} Qx\varphi$. Let $S(s)$ be the set of *exactly all* assignments $s[a/x]$ ($a \in A$) such that $\mathfrak{A}, s[a/x] \models_{\text{FO}} \varphi$. As $\mathfrak{A}, s \models_{\text{FO}} Qx\varphi$, we have $A(s) := \{a \in A \mid s[a/x] \in S(s)\} \in Q^A$. Notice that $A \setminus A(s) \in Q'^A$.

Define the function $f : U \rightarrow Q^A$ such that $f(s) = A(s)$ for each $s \in U$. By Proposition 2.1, we have $\mathfrak{A}, U[f/x] \models \varphi$. Let $g : U \rightarrow \mathcal{P}(A)$ be the function such that $g(s) = A \setminus f(s)$ for all $s \in U$.

Since $\mathfrak{A}, U[f/x] \models \varphi$, we have $\mathfrak{A}, U[f/x] \models \varphi^*$ by the induction hypothesis. Since f maps from U to Q^A , we have $\mathfrak{A}, U[f/x] \models A_Q(\overline{y}, x)$. Thus $\mathfrak{A}, U[f/x] \models A_Q(\overline{y}, x) \wedge \varphi^*$. To show that $\mathfrak{A}, U \models (Qx\varphi)^*$, it we must prove that

$$\mathfrak{A}, U[A/x] \models (A_Q(\overline{y}, x) \wedge \varphi^*) \vee A_{Q'}(\overline{y}, x). \quad (4)$$

Since $U[f/x] \cup U[g/x] = U[A/x]$, it now suffices to show that $\mathfrak{A}, U[g/x] \models A_{Q'}(\overline{y}, x)$. Recall that $A \setminus A(s) \in Q'^A$ for each $s \in U$. Thus, by the definition of g , we have $\mathfrak{A}, U[g/x] \models A_{Q'}(\overline{y}, x)$. Therefore $\mathfrak{A}, U[g/x] \models A_{Q'}(\overline{y}, x)$.

Assume then that $\mathfrak{A}, U \models (Qx\varphi)^*$. Therefore Equation 4 holds. We assume that $U \neq \emptyset$, for otherwise we are done. Choose an arbitrary assignment $s \in U$. We will show that $\mathfrak{A}, \{s\} \models Qx\varphi$. Due to Proposition 2.1, and since we choose s arbitrarily, this will directly imply that $\mathfrak{A}, U \models Qx\varphi$.

Since Equation 4 holds, we infer that $\mathfrak{A}, V \models A_Q(\overline{y}, x) \wedge \varphi^*$ and $\mathfrak{A}, X \models A_{Q'}(\overline{y}, x)$ for some teams V and X such that $V \cup X = U[A/x]$.

Let V_{-x} denote the team with domain $\text{Dom}(V) \setminus \{x\}$ such that for all assignments t , we have $t \in V_{-x}$ iff there exists some element $a \in A$ such that $t[a/x] \in V$. Assume first that $s \in V_{-x}$, where s is the assignment in U we appointed above. As $\mathfrak{A}, V \models A_Q(\overline{y}, x)$, we see that there is a set $B \in Q^A$ such that $s[B/x] \subseteq V$. As $\mathfrak{A}, V \models \varphi^*$, we have $\mathfrak{A}, V \models \varphi$ by the induction hypothesis. Therefore, using Proposition 2.1, we infer that $\mathfrak{A}, s[B/x] \models \varphi$. Thus $\mathfrak{A}, \{s\} \models Qx\varphi$, as desired.

Assume then that $s \notin V_{-x}$. Thus $s \in X_{-x}$, where X_{-x} is of course the team with domain $\text{Dom}(X) \setminus \{x\}$ such that for all assignments t , we have $t \in X_{-x}$ iff there is some $a \in A$ such that $t[a/x] \in X$. Let S be the set of all assignments $t \in X$ such that there is some $a \in A$ such that $t = s[a/x]$. Since $s \notin V_{-x}$ and $V \cup X = U[A/x]$, we have $S = s[A/x]$. As $\mathfrak{A}, X \models A_{Q'}(\overline{y}, x)$, we therefore see that $A \in Q'^A$. Thus $\emptyset \in Q^A$. Thus, as Q is monotone, we have $Q^A = \mathcal{P}(A)$. Thus trivially $\mathfrak{A}, s \models_{\text{FO}} Qx\varphi$, whence $\mathfrak{A}, \{s\} \models Qx\varphi$ by Proposition 2.1. \square

5 Conclusions

We have shown how the standard logics based on team semantics extend naturally to the simple system D^* that captures RE. The system D^* nicely expands the scope of team semantics from logic to computation. It will be interesting to investigate, for example, *what kind of decidable fragments D^* has*.

References

- [1] Egon Börger and Robert F. Stärk. *Abstract State Machines. A Method for High-Level System Design and Analysis*. Springer, 2003.
- [2] Fredrik Engström. Generalized quantifiers in dependence logic. *Journal of Logic, Language and Information*, 21(3):299–324, 2012.
- [3] Fredrik Engström and Juha Kontinen. Characterizing quantifier extensions of dependence logic. *J. Symb. Log.*, 78(1):307–316, 2013.
- [4] Fredrik Engström, Juha Kontinen, and Jouko A. Väänänen. Dependence logic with generalized quantifiers: Axiomatizations. In *WoLLIC*, pages 138–152, 2013.
- [5] Pietro Galliani. Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. *Ann. Pure Appl. Logic*, 163(1):68–84, 2012.
- [6] Pietro Galliani and Lauri Hella. Inclusion logic and fixed point logic. In *CSL 2013*, pages 281–295, 2013.
- [7] Erich Grädel and Jouko Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013.
- [8] Y. Gurevich. A new thesis. *American Mathematical Soc. Abstracts*, 6(4):317, 1985.
- [9] J. Hintikka and G. Sandu. Informational independence as a semantical phenomenon. In *Logic, Methodology and Philosophy of Science VIII*. Elsevier, 1989.
- [10] W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL*, 5:539–563, 1997.
- [11] Antti Kuusisto. A double team semantics for generalized quantifiers. *CoRR*, abs/1310.3032, 2013.
- [12] Antti Kuusisto. Some Turing-complete extensions of first-order logic. *CoRR*, abs/1405.1715, 2014.
- [13] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.
- [14] Jouko Väänänen. *Dependence logic: A new approach to independence friendly logic*. Cambridge University Press, 2007.